Memorandum 6M-4789-1

Sheet 1 of 19 Sheets

Division 6 — Lincoln Laboratory
Massachusetts Institute of Technology
Lexington 73, Massachusetts

SUBJECT: A FUNCTIONAL DESCRIPTION OF THE TX-0 COMPUTER

To: Distribution List

From: J.T. Gilmore, Jr. and H.P. Peterson [illegible]

Date: October 3, 1958

Approved: [illegible]

Abstract: The TX-0 is an experimental digital computer which was constructed to check transistor circuitry and a 256 x 256 magnetic core memory. The logical design is rather simple since it has only four instructions. Three of these refer to memory in the normal way, but the fourth has the interesting feature of providing the facility to micro-program via time pulses. How useful this is will be determined by the experience gained in programming for TX-0. This memo has been written to give the reader a working knowledge of the computer's logic, usefulness, and capabilities.

Distribution List:

Group 63 Staff
Arden, Dean, 26-269
Arnow, J., Gr. 28
Attridge, W.
Bagley, P.R., Gr. 67
Bailey, D., Gr. 67
Briscoe, H., Gr. 67
Buzzard, R., Gr. 64
Daggett, N., Gr. 64
Dinneen, G.P., Gr. 32
Dustin, D.E., Gr. 312
Frachtman, H.E., Gr. 67
Grandy, C., Gr. 64
Hazel, F.P., Gr. 60
Heart, F., Gr. 21
Holmes, L., Gr. 64
Israel, D., Gr. 66
Rich, E., Gr. 64
Rising, H.K., Gr. 67
Thomas, L.M., Gr. 67
Tritter, A.L., Gr. 34
Vance, P.R., Gr. 61
Zraket, C.A., Gr. 67

JTG,HPP:med

2. The research reported in this document was supported jointly by the Department of the Army, the Department of the Navy, and the Department of the Air Force under Air Force Contract No AF 19 (122)-458.

## TABLE OF CONTENTS

## I. INTRODUCTION

The TX-0 computer is a general purpose high-speed machine built primarily of transistors. The computer has one memory device which is a vacuum-tube-driven magnetic-core array capable of storing 1,179,648 bits of information. Each word contains 18 bits for a total of 65,536 or $2^{16}$ words.(1) The memory cycle time is approximately 6 μsec. The machine performs a complete operation every two memory cycles; the instruction is obtained in the first cycle and the data in the second.(2) Most of the logical and arithmetic operations are executed in the second cycle.

## II PRESENT TERMINAL EQUIPMENT

Input

1. A Ferranti photoelectric paper tape reader
   - (a) Standard seven-hole flexowriter paper tape
   - (b) 200 to 250 lines per second
2. Toggle switch registers
   - (a) Toggle switch accumulator called TAC
   - (b) Toggle switch buffer register called TBR
   - (c) 16 toggle switch registers called toggle switch storage, TSS. These registers can replace the first 16 registers of magnetic core memory by means of a switch on the main console.
3. Flexoprinter input to live register bits 2, 5, 8, 11, 14, 17 setting $LR_0$ to a one when the key is struck.
4. Provision has been made for a photosensitive device, called the light pen, to control the computer from the display tube.

Output

1. One 12 1/2" cathode ray oscilloscope display tube
   - (a) 511 points by 511 points in 7" by 7" array
   - (b) a camera will be added in the future
2. Paper tape punch

---

1. The TX-2 which is in the process of being constructed will use this memory. The TX-0 will then have a transistor-driven core memory of $2^{12}$ registers.

2. Each memory cycle has eight time pulses and the notation we use in referring to them is cycle, time pulse, (i.e., cycle 0, time pulse 8 is written 0.8).

(a) Standard flexowriter tape

3. Standard flexowriter printer

## III REGISTERS

1. Memory Buffer Register - (MBR, 18 bits + 1 parity check bit) receives information from and sends information to the memory. The transfer of information from the memory is checked by means of the parity digit which makes the sum of all 19 digits odd.

2. Accumulator - (AC, 18 bits) - stores the results of numerical operations - is also used as buffer to in-out terminal equipment. The bits of AC are numbered from left to right, 0 to 17.

   One interesting point with regard to the AC is that one may look upon it as strictly a ring adder. If we consider the left-most digit as a sign, then the largest representable number is $2^{17}-1$ and the smallest is $-2^{17}+1$. If a one is added to the largest number the result is the smallest and likewise if a one is subtracted from the smallest the result is the largest. There is no overflow alarm. This feature has already been found to be useful in decision techniques.

3. Memory Address Register - (MAR, 16 bits) - selects the information in the memory and has another special feature of selecting operate class commands - (more about this later).

4. Program Counter - (PC, 16 bits) - is used by control and contains the address of the next instruction to be executed.

5. Instruction Register - (IR, 2 bits) - contains the operation part of the instruction which is to be executed.

6. Live Register - (LR, 18 bits) - may be considered as just another storage register which uses flip-flop rather than magnetic cores. It is referred to by means operate class commands which we shall see later.

7. Toggle Switch Buffer Register - (TBR, 18 toggle switches) - used for manual intervention in the normal and test modes.

8. Toggle Switch Accumulator - (TAC, 18 toggle switches) - used for manual intervention in the normal and test modes.

For a description of the flip-flops and logical control, see Figure 6.

## IV INSTRUCTIONS AND OPERATING MODES

The first two bits of the 18 bit TX-0 word designate one of four basic instructions. The machine recognizes which one to perform by means of two flip-flops $IR_0$ and $IR_1$ called the instruction register. The remaining 16 bits of three of the instructions are used to specify a memory location. The fourth instruction makes use of its remaining 16 bits to designate one or more special commands. These are called operate class commands and are the means by which TX-0 attains its versatility. (As we shall see in section VIII and IX).

TX-0 has three operating modes: Normal, Test, and Read-In. They are specified by two flip-flops, R and T called the mode register. The four instructions are carried out in one of the three modes and for each of the twelve combinations a different function is executed by the machine. The console has a push button to select the Test mode and also one for the Read-In mode. The Normal mode is initiated by instructions in the other two modes. In the Normal mode instruction words are taken from the stored program; in the Test mode, from the TBR; and in the Read-In mode, from the tape being read in.

The mode register (R and T) decodes the modes as follows:

| MODE | R | T |
|---|---|---|
| Normal | 0 | 0 |
| Test | 0 | 1 |
| Read-In | 1 | 1 |

## V THE NORMAL MODE

The four basic instructions in the Normal mode are interpreted as follows:

| $IR_0$ $IR_1$ | ABBREVIATION | INSTRUCTION |
|---|---|---|
| 0 0 | sto x | Replace the contents of register x with the contents of the AC. Let the AC remain the same. |
| 0 1 | add x | Add the word in register x to the contents of the AC and leave the sum in the AC. |
| 1 0 | trn x | If the sign digit of the accumulator ($AC_0$) is negative (i.e., a one) take the next instruction from register x and continue from there. If the sign is positive (i.e., a zero) ignore this instruction and proceed to the next instruction. |
| 1 1 | opr x | Execute one of the operate class commands indicated by the number x. (See sections VIII and IX). |

## VI TEST MODE

The test mode is selected by a push button on the console. Primarily the Test mode was designed into the computer to aid engineers and operators to manually intervene with control and storage for test purposes.

Basically one may consider the test mode as being a one instruction program where the instruction is set in the TBR (Toggle Switch Buffer) and the data to be treated either already in the AC or set in the TAC (Toggle Switch AC). There are two switches on the console which allow a little more versatility to the one instruction. They are called the repeat and step switches. The repeat switch causes the instruction to be repeated over and over again (unless, of course, it is of the transfer control type). The step switch allows the address section of the instruction to be indexed by one each time the instruction is executed.

When the test mode push button on the console is activated (i.e., pushed) the first two digits of the TBR are sent to the IR and the last 16 digits are sent to the MAR. (In the sto x case the AC is reset according to what is set in the TAC). The PC is set to MAR + 1 and the instruction is executed.

Then if:

| Repeat Switch | Step Switch | Operation After Execution of the Instruction |
|---|---|---|
| Off | Off | The computer will stop |
| Off | On | The computer will stop but the MAR will be changed to what is in the PC namely, the preceding MAR + 1 and then the PC will be indexed by 1. |
| On | Off | The computer will continue to perform the same instruction repeatedly at machine speed. |
| On | On | The MAR will be changed to what is in the PC, namely the preceding MAR + 1. Then the PC will again be indexed by 1 and the instruction will be executed repeatedly with the address section being stepped up by one each time. |

The four basic instructions for the test mode are classified as load, examine, test operate, and start.

| | | |
|---|---|---|
| "Load" | sto x<br>0 0 | The AC is set to what is in the TAC and then the contents of the AC are stored in register x. |

| | | |
|---|---|---|
| "Examine" | add x<br>0 1 | The contents of register x are added to the AC by means of the MBR. Hence x can be examined in the MBR. The AC could have been"anything" before the instruction so all we can say is that the AC will contain "anything" plus the contents of x. |
| "Test Operate" | opr x<br>1 1 | Any one of the operate class commands is executed. Stepping means nothing in this instruction |
| "Start" | trn x<br>1 0 | Change to normal mode and transfer control to instruction in register x. Stepping and repeating mean nothing in this in-struction. |

## VII <u>READ-IN MODE</u>

The Read-In mode is selected by a push button on the console and causes the photoelectric reader to be activated. As each line of tape passes under the read head, the information in tape positions 1, 2, 3, 4, 5, and 6 is transferred to digital positions 0, 3, 6, 9, 12, and 15 of the AC. Once the first line of information is in the AC, the AC is cycled to the right one digital position. The second line is then read in, the AC cycled again one position, and the third line read in. At this point the first three lines are now assembled as a word in the AC. The tapes to be used by the Read-In mode have been made so that each word to be stored follows an instruction word on tape which will perform the storage. In order to transfer control to inner storage all that is required on tape is the transfer instruction itself and it will not be followed by the usual three lines of data as the store instruction is. Getting back to the mechanics of the read-in, the first three lines of information have been read in and assembled in the AC. Since this word will be an instruction in either the storage or the transfer case, the first two digits in the AC are transferred to the IR (Instruction Register) and the last 16 digits to the MAR. At this point the instruction register is examined and if the instruction is of the storage type then the next three lines of tape are read in and assembled in the AC and then the instruction is executed. If when the IR was examined the instruction was of the transfer control type then no more information is read in and the transfer control instruction is executed. Summarizing, we can say that each data word requires six lines of tape; the first three indicating where to put it and the last three the word itself; each transfer control instruction requires only three lines of tape containing the instruction itself. The tape layout can be seen more clearly in Figure 5.

The four basic instructions of the Read-In mode are separated into two types - storage and transfer control.

| Type | First IR | Modified IR | Lines of tape | Symbol | Description |
|---|---|---|---|---|---|
| Storage | 0 0 | 0 0 | 6 | sto x | Store the word (which was read in behind this instruction) in register x. |
| Storage | 1 1 | 0 0 | 6 | opr x | When the two digits for opr x (11) are read into IR (in the Read-In mode) they are complemented and therefore opr x = sto x. |
| Transfer Control | 1 0 | 1 0 | 3 | trn x | When the two digits of trn (10) are read into IR (in the Read-In mode) the computer stops reading from tape; the computer is changed to the normal mode and control is immediately transferred to register x. |

| Type | First IR | Modified IR | Lines of tape | Symbol | Description |
|---|---|---|---|---|---|
| Transfer | 0 1 | 1 0 | 3 | add x | When the two digits of add (01) are read into IR (in the Read-In mode) they are complemented and the computer stops. Upon restarting (by pushing the restart button on the console) the computer performs the instruction trn x. Therefore, add x = stop + trn x. |

Note, that with a hand punch, instruction-words on the tape can be modified so that st (00) can become add (01) or transfer (10) and either add or transfer can become operate (11). The flexibility allows changes on the tape without preparing a new one.

In actual practice the Read-In mode is used to read a more efficient Read-In program into storage, since a binary tape with a store instruction following each word would be extremely large and cumbersome. A description of this program will be found in section X. It is called the Input Routine and further describes how data is put into storage and also gives the reader an example of TX-0 programming.

## VIII OPERATE CLASS COMMANDS

The following is a list of the operate class commands, the time pulse on which they are executed, the binary form they assume, what they do and the octal notation of the last 16 bits of the operate instruction, opr x.

| IR | | MAR | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | |
| | | <u>2</u> | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = opr 100,000 (octal) |
| (0.8) CLL = Clear the left nine digital positions of the AC | | | | | | | | | | | | | | | | | | |
| | | | <u>3</u> | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = opr 40,000 (octal) |
| (0.8) CLR = Clear the right nine digital positions of the AC | | | | | | | | | | | | | | | | | | |
| | | | | <u>4</u> | <u>5</u> | | | | | | | | | | | | | |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = opr 20,000 (octal) |
| (0.8) IOS In-Out Stop = Stop machine so that an In-Out command (specified by digits 6 7 8 of MAR) may be executed. | | | | | | | | | | | | | | | | | | |
| | | | | <u>4</u> | <u>5</u> | | | | | | | | | | | | | |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = opr 30,000 (octal) |
| (1.8) Hlt = Halt the computer | | | | | | | | | | | | | | | | | | |
| | | | | | | <u>6</u> | <u>7</u> | <u>8</u> | | | | | | | | | | |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = opr 7,000 (octal) |
| (0.8) P7H = Punch holes 1-6 in flexo tape specified by AC digital positions 2, 5, 8, 11, 14, and 17. Also punch a 7th hole on tape. | | | | | | | | | | | | | | | | | | |
| | | | | | | <u>6</u> | <u>7</u> | <u>8</u> | | | | | | | | | | |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = opr 6,000 (octal) |
| (0.8) P6H = Same as P7H but no seventh hole | | | | | | | | | | | | | | | | | | |

```
 IR                               MAR
0  1   2   3  4  5   6  7  8   9  10  11   12  13  14   15  16  17
```

---

```
                     6  7  8
1  1   0   0  0  0   1  0  0   0  0   0    0   0   0    0   0   0 = opr 4,000 (octal)
```

(0.8) PNT = Print one flexowriter character specified by AC digits 2, 5, 8, 11, 14, and 17.

---

```
                     6  7  8
1  1   0   0  0  0   0  0  1   0  0   0    0   0   0    0   0   0 = opr 1,000 (octal)
```

(0.8) R1C = Read one line of flexo tape so that tape positions 1, 2, 3, 4, 5, and 6 will be put in the AC digital positions 0, 3, 6, 9, 12 and 15.

---

```
                     6  7  8
1  1   0   0  0  0   0  1  1   0  0   0    0   0   0    0   0   0 = opr 3,000 (octal)
```

(0.8) R3C = Read one line of flexo tape into AC digits 0, 3, 6, 9, 12 and 15. Then cycle the AC one digital position; read the next line on tape into AC digits 0, 3, 6, 9, 12 and 15, cycle the AC right one digital position and read the third and last line into AC digits 0, 3, 6, 9, 12 and 15. (This command is equal to a triple CYR-R1C.)

---

```
                     6  7  8
1  1   0   0  0  0   0  1  0   0  0   0    0   0   0    0   0   0 = opr 2,000 (octal)
```

(0.8) DIS = Intensify a point on the scope with x and y coordinates where x is specified by AC digits 0-8 with digit 0 being used as the sign and y is specified by AC digits 9-17 with digit 9 being used as the sign for y. The complement system is in effect when the signs are negative.

---

```
                               9  10
1  1   0   0  0  0   0  0  0   1  0   0    0   0   0    0   0   0 = opr 400 (octal)
```

(1.4) SHR = Shift the AC right one place, i.e., multiply the AC by $2^{-1}$

---

```
                               9  10
1  1   0   0  0  0   0  0  0   1  1   0    0   0   0    0   0   0 = opr 600 (octal)
```

(1.4) CYR = Cycle the AC right one digital position ($AC_{17}$ will become $AC_0$)

---

```
                               9  10
1  1   0   0  0  0   0  0  0   0  1   0    0   0   0    0   0   0 = opr 200 (octal)
```

(1.3) MLR = Store the contents of the MBR (memory buffer register) in the live reg.

```
 IR                                       MAR
  1   2   3  4  5   6  7  8   9  10  11   12  13  14   15  16  17
------------------------------------------------------------------------------
                                      11                15
. 1   0   0  0  0   0  0  0   0  0    1    0   0   0    0   0   0 = opr 100 (octal)
```

1.1) PEN = Read the light pen flip-flops 1 and 2 into $AC_0$ and $AC_1$.

```
                                      11                15
. 1   0   0  0  0   0  0  0   0  0    0    0   0   0    1   0   0 = opr 4 (octal)
```

1.1) TAC = Insert a one in each digital position of the AC wherever there is a one in the corresponding digital position of the TAC.

```
                                           12
. 1   0   0  0  0   0  0  0   0  0    0    1   0   0    0   0   0 = opr 40 (octal)
```

1.2) COM = Complement every digit in the accumulator

```
                                               13
. 1   0   0  0  0   0  0  0   0  0    0    0   1   0    0   0   0 = opr 20 (octal)
```

1.4) PAD = Partial add AC to MBR, that is, for every digital position of the MBR that contains a one, complement the digit in the corresponding digital position of the AC. This is also called a half add.

```
Example:   AC = 1  0  1  0  1  0  1
          MBR = 0  1  1  1  0  0  0
          ---------------------------
   New     AC = 1  1  0  1  1  0  1
```

```
 IR                          MAR
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17
```

```
                                           14
1  1  0  0  0  0  0  0  0  0  0  0  0  0   1  0  0  0 = opr 10 (octal)
```

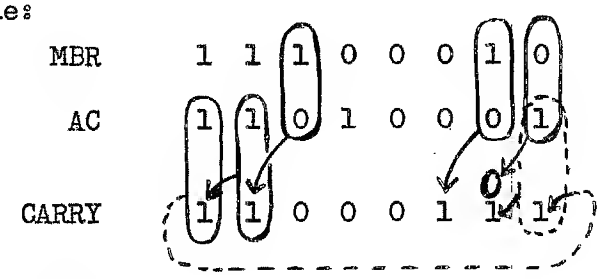
(1.7) CRY = Partial add the 18 digits of the AC to the corresponding 18 digits of the carry.

To determine what the 18 digits of the carry are, use the following rule:

"Grouping the AC and MBR digits into pairs and proceeding from right to left, assign the carry digit of the next pair to a one if in the present pair MBR = 1 and AC = 0 or if in the present pair AC = 1 and carry 1.

(Note: The 0th digit-pair determines the 17th pair's carry digit)

Example:



```
CARRY    1  1  0  0  0  1  1  1

   AC    1  1  0  1  0  0  0  1
        ______________________
New AC   0  0  0  1  0  1  1  0
```

---

```
                                                 16 17
1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 = opr 1
```

(1.2) AMB = Store the contents of the AC in the MBR.

---

```
                                                 16 17
1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1 = opr 3
```

(1.2) TBR = Store the contents of the TBR in the MBR.

---

```
                                                 16 17
1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0 = opr 2
```

(1.3) LMB = Store the contents of the LR in the MBR.

It should be noticed that the command CYL or cycle left was not listed. The reason for that is:

$$\text{CYL} = \overset{(1.2)}{\text{AMB}},\ \overset{(1.4)}{\text{PAD}},\ \overset{(1.7)}{\text{CRY}}$$

<u>Example</u>

| | | |
|---|---|---|
| After AMB (1.2) | AC = | 0 0 1 1 0 0 1 1 |
| | MBR = | 0 0 1 1 0 0 1 1 |
| After PAD (1.4) | MBR = | 0 0 1 1 0 0 1 1 |
| | AC = | 0 0 0 0 0 0 0 0 |
| | CARRY = | 0 1 1 0 0 1 1 0 |
| After CRY (1.7) | AC = | 0 1 1 0 0 1 1 0 |

This is an excellent example of how a programmer can accomplish many things with one operate instruction. Also notice that the AC was cleared by AMB + PAD. Any operate instruction-word is capable of having a large variety of commands within itself as long as the programmer is aware of the time pulse sequence. The preceding list of commands also lists the cycle and time pulse of each command. We have included a TX-O logical flow chart in this memo with a few side remarks on the chart to assist you in reading it. Whenever there is some question as to what is happening on each time pulse this chart should give you the answer. If you find in experimenting with the operate class commands that a new operate class command would be useful to a programmer, we will be glad to consider your suggestions.

## IX COMBINATIONS OF OPERATE CLASS COMMANDS

The following list of combinations has already been found to be useful in programming. A conversion program which will be described in a later memo is capable of assembling most of these combinations using three letter mnemonic symbols (e.g., lac, alr, lad, etc.)

```
0.8   0.8
CLL + CLR = opr 140,000 = clear the AC (CLA)

1.2   1.4   1.7
AMB + PAD + CRY = opr 31 = cycle the AC left one digital position (CYL)

0.8   0.8   1.2
CLL + CLR + COM = opr 140,040 = clear and complement AC (CLC)

0.8   0.8
IOS + DIS = opr 22,000 = Display (this combination was included to remind
            you that with every in-out command the IOS must be included)
            (DIS)

0.8   0.8   0.8
IOS + CLL + CLR = opr 160,000 = In out stop with AC cleared.

0.8   0.8   1.4
IOS + P7H + CYR = opr 27,600 = Punch 7 holes and cycle AC right.

0.8   0.8   1.4
IOS + P6H + CYR = opr 26,600 = Punch 6 holes and cycle AC right.

0.8   0.8   0.8   0.8
IOS + CLL + CLR + P6H = opr 166,000 = Clear the AC and punch a blank
                        space on tape.

0.8   0.8   0.8
IOS + PNT + CYR = opr 24,600 = Print and cycle AC right.

0.8   0.8   1.2   1.4
IOS + P7H + AMB + PAD = opr 27,021 = Punch 7 holes and leave AC cleared.

0.8   0.8   1.2   1.4
IOS + P6H + AMB + PAD = opr 26,021 = Punch 6 holes and leave AC cleared.

0.8   0.8   1.2   1.4
IOS + PNT + AMB + PAD = opr 24,021 = Print and leave AC cleared.

0.8   0.8   0.8
CLL + CLR + RIC = opr 141,000 = Clear AC and start petr running (notice
                  no IOS - which means computer hasn't stopped to wait
                  for information).
```

```
0.8   1.2   1.4   1.7
RIC + AMB + PAD + CRY = opr 1,031 = Start petr running and cycle AC left

0.8   0.4
RIC + CYR = opr 1,600 = Start petr running and cycle right.

0.8   0.8   0.8   0.8
CLL + CLR + IOS + R3C = opr 163,000 = Clear AC, read 3 lines of tape.

0.8   0.8   0.8   0.8
CLL + CLR + IOS + RIC = opr 161,000 = Clear AC and read one line of tape.

0.8   0.8   0.8   0.8   1.4   1.7
CLR + CLR + IOS + RIC + PAD + CRY = opr 161,031 = Read 1 line of tape and
                                                  cycle AC left.

0.8   0.8   0.8   0.8   1.4
CLL + CLR + IOS + RIC + CYR = opr 161,600 = Read one line of tape and
                                            cycle right.

0.8   0.8   1.1
CLL + CLR + TAC = opr 140,004 = Put contents of TAC in AC.

1.4   1.7
PAD + CRY = opr 30 = Full-add the MBR and AC and leave sum in AC.

0.8   0.8   1.3   1.4
CLL + CLR + LMB + PAD = opr 140,022 = Clear the AC - store LR contents in
                                      memory buffer register - add memory
                                      buffer to AC - i.e., store live reg.
                                      contents in AC. (LAC)

1.2   1.3
AMB + MLR = opr 201 = Store contents of AC in MBR, store contents of MBR
                      in LR, i.e., store contents of AC in LR. (ALR)

1.3   1.4
LMB + PAD = opr 22 = Store contents of LR in MBR, partial add AC and MBR
                     i.e., partial add LR to AC. (LPD)

1.3
MLR = opr 200 = Since MLR alone will have a clear MBR, this is really
                clear LR. (LRO)

1.3   1.4   1.7
LMB + PAD + CRY = opr 32 = Full-add the LR to the AC. (LAD)

0.8   0.8   1.3   1.4
CLL + CLR + TBR + PAD = opr 140,023 = Store contents of TBR in AC.
```

## X PROGRAM EXAMPLE

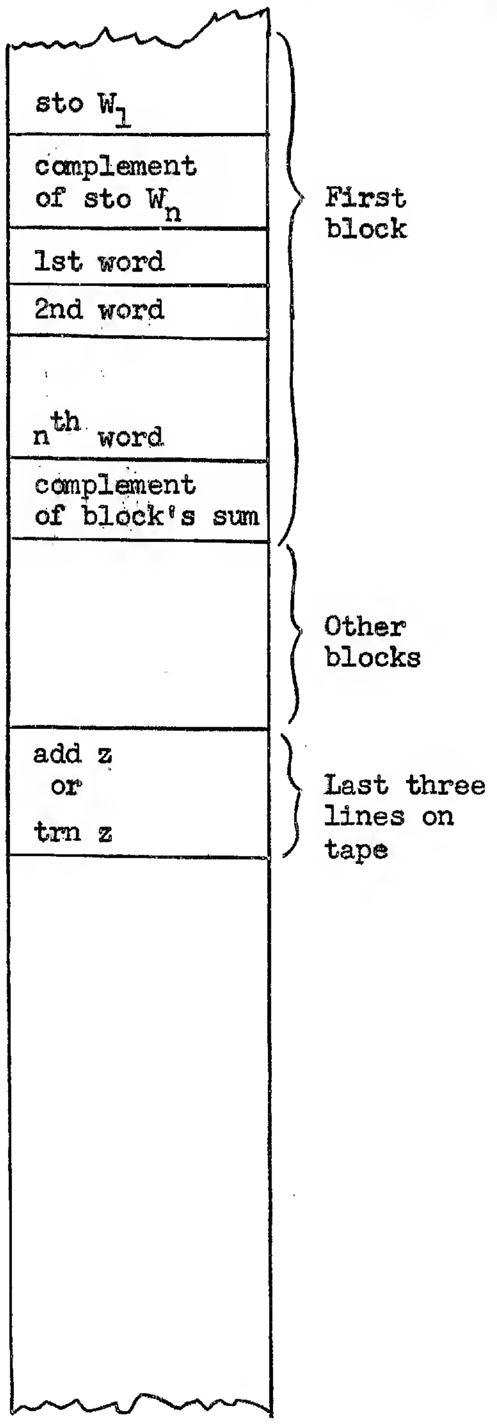
This section was included to give the reader an example of a TX-0 program. The program which was chosen is used to read binary tapes into storage and is called the Input Routine. It was written to avoid the long and cumbersome tapes which would be required by the Read-In mode (a store instruction for each data word). When the conversion program has finished converting a program's flexowriter tape and is ready to punch a binary tape, it first punches the Input Routine on tape in the form that is required by the Read-In mode. Then the converted program is punched out in binary form according to the specifications required by the Input Routine.

By having the Input Routine on the leader of each tape all that is required is the activation of the Read-In push button. The Input Routine is read in by the Read-In mode and then control is immediately transferred to the Input Routine which takes on the task of reading in the rest of the binary tape.

The specifications required by the Input Routine are very simple. The tape channel positions of a word are the same as they are in the Read-In mode. Words are transferred to storage in blocks of sequentially addressed words. The first word in the block is a store instruction word whose address section contains the address of the first data word in the block. (Call it sto $W_1$.) The second word in the block is the complement of a store instruction word whose address contains the address of the last data word in the block. (Call it sto $W_n$ where n = no. of data words.) The data words follow these two pieces of information. Following the last data word of the block is a word which is the complement of the sum of all the preceding words in the block including the first two control words.

The address of the starting instruction follows the last block of data words. If it is in the form of an add instruction (add z) the computer will be stopped before the Input Routine transfers control to the program. If it is in the form of a transfer control instruction (trn z) then the program will be started immediately after the last block of data words has been read into storage.

TAPE FORMAT REQUIRED BY INPUT ROUTINE



$W_1$ = Address of 1st data word in block

$W_n$ = Address of nth data word

trn z = Upon completion of last block transferred to storage take first instruction from register z.

Add z = Stop computer after last block has been transferred to storage. After restart, take first instruction from register z.

## INPUT ROUTINE

| | | | |
|---|---|---|---|
| | 177741 | Temporary storage | Partial sum of block |
| | 177742 | Add 177773 | If the preceding block's sum is correct, go on to next block or transfer control word. If not, go to 177772 and stop computer. |
| | 177743 | trn 177772 | |
| Enter | 177744 | opr 163,000 (R3C) | Read in the first word of a block or the transfer control word (add Z or trn Z) and store it in register 177756. |
| | 177745 | sto 177756 | |
| | 177746 | trn 177756 | Is it st $W_1$, add Z, or trn Z? If trn Z go directly to register 177756. |
| | 177747 | add 177774 | It is either st $W_1$ or add Z; add 200,000 to the AC. If it was add Z, the AC is now neg. (=trn Z), so go to 177775. |
| | 177750 | trn 177775 | |
| | 177751 | opr 163,000 (R3C) | Read in the complement of the address of the last word in the block and store it in the register 177777. |
| | 177752 | sto 177777 | |
| | 177753 | add 177756 | Add the first two control words of the block together and store in 177741 to initiate the partial sum. |
| | 177754 | sto 177741 | |
| | 177755 | opr 163,000 (R3C) | Read in the i th word and store it in its assigned memory location |
| | 177756 | (sto Wi) (add Z) (Not used) (trn Z) | |
| | 177757 | add 177741 | Add the i th word to the partial sum of the block. |
| | 177760 | sto 177741 | |
| | 177761 | opr 140,000 (CLA) | Index the address section of the register 177756 by one. |
| | 177762 | add 177756 | |
| | 177763 | add 177773 | |
| | 177764 | sto 177756 | |
| | 177765 | add 177777 | Has the n th word been transferred to storage? If AC is negative - no, return to 177755. |
| | 177766 | trn 177755 | |
| | 177767 | opr 163,000 (R3C) | Read in the sum of the block. Is it the same as the sum in register 41? If it is, AC = minus zero, go to 177742. |
| | 177770 | add 177741 | |
| | 177771 | trn 177742 | |
| | 177772 | opr 30,000 (HLT) | If it is positive.... stop the computer. The sum check is wrong. |
| | 177773 | 1 | Constants |
| | 177774 | 200,000 | |
| | 177775 | sto 177777 | The last block has been stored and the transfer control word was add Z. Put trn Z in register 177777 and stop the computer. |
| | 177776 | opr 30,000 (HLT) | |
| | 177777 | trn Z (or the complement of the address of the last word in a block) | Upon restarting, transfer control to register Z. |

The operate class commands used in the Input routine were:

opr 160,000 = CLL + CLR + IOS + R3C
Clear AC and read three lines, cycling each time so that they are assembled as an 18 bit word in the AC.

opr 140,000 = CLL + CLR = CLA
Clear both halves of AC.

opr 30,000 = Halt the computer

It should be noticed that a trn instruction (10) has a one in the sign digital position. In registers 177744, 45, and 46 when the transfer control word "trn Z" is read into the accumulator, the trn 177756 will transfer control to 177756. Since register 177756 will contain trn Z and the AC still contains trn Z, control will immediately be sent to register Z. This is a useful trick. (For example, transferring control to a subroutine with the exit word in the AC).

One other point of interest, if the word add Z is in the AC when instruction trn 177756 in register 177746 is performed, the AC is positive and the next instruction will be add 177774. This will cause the octal number 200,000 to be added to the AC and since the first two bits of the word add Z are 01, the result will be trn Z. This causes the instruction trn 177775 to be executed and 177775 will store the word trn Z in register 177777. The next instruction is the operate class command halt. Since the AC is not disturbed, it will still contain trn Z. If the restart push button is activated, the trn Z in register 177777 will transfer controll to register Z.

## XI TOGGLE SWITCH STORAGE

The TX-0 has an auxiliary memory system consisting of sixteen toggle-switch registers which we shall refer to as toggle switch storage, TSS. The TSS can be used as a substitute for the first sixteen magnetic core registers 0 through 17. All sixteen registers of TSS can replace core registers 0 through 17 or they can be chosen individually to replace their respective core registers, i.e., $TSS_6$ can replace register 6 of core memory while the other fifteen can still be core.

The Live Register has been mentioned earlier as an eighteen bit flip-flop register with no address. Up to this point the only way reference could be made to it was by means of the operate class commands. The switches on the TSS panel allow the Live Register to be addressed like any other register. However, its contents can still only be changed by specific operate class commands or by data from the flexo typewriter (if the flexo input switch on the main console is in the on position.)

The sixteen registers of TSS are located on the console. (See Figure 4) In addition to the eighteen toggle switches associated with each register there is a toggle switch located to the left of each register which we shall call "cm" and one to the right of each register which we shall call "lr". Also located on the console is a master switch called "core memory select" or CM select. When the CM select switch is on, the first sixteen registers will <u>always</u> be magnetic core. When the CM select switch is <u>off</u> then the first sixteen registers can be either magnetic core, toggle switch storage, or addresses of the live register.

The following is a breakdown of the possible combinations:

CM Select Switch = OFF

| Reg. | cm | $TSS_x$ | lr | |
|---|---|---|---|---|
| | | | | <u>Case One</u> |
| x | Off | W | Off | Register x is TSS and the word in x is W which is set by the toggle switches. |
| | | | | <u>Case Two</u> |
| x | Off | W | On | x is the address of the LR and the word in x will always be the word in the LR and not the toggle sw setting W. |
| | | | | <u>Case Three</u> |
| x | On | W | Off | Register x is magnetic core and the toggle switch setting W means nothing. |
| | | | | <u>Case Four</u> |
| x | On | W | On | The core switch cm takes precedence over the lr switch and this case becomes the same as case three. |

Note that the Live Register may have one, two, three or sixteen different addresses (0 - 17) or none at all if no lr switch is on.

Attachments:

Appendix A
Appendix B
Appendix C

| | |
|---|---|
| Fig. 1 | A-68266 |
| Fig. 2 | A-68264 |
| Fig. 3 | A-68265 |
| Fig. 4 | A-68263 |
| Fig. 5 | A-68405 |
| Fig. 6 | E-69059 |
| Fig. 7 | D-47243 |

# APPENDIX A

## TX-0 Console

1. <u>Push Buttons</u>

   (a) Stop
   (b) Restart
   (c) Read-In
   (d) Test
   (e) Tape Feed

2. <u>Flip-Flop Indicators</u>

| | | |
|---|---|---|
| (a) | IR | Two bit instruction register |
| (b) | C | Cycle |
| (c) | RT | Mode |
| (d) | MR | Memory Read |
| (e) | MI | Memory Inhibit |
| (f) | PAR | Parity |
| (g) | SS | Start Stop |
| (h) | PBS | Push Button Synchronizer |
| (i) | IOS | In Out Stop |
| (j) | CH | Chime Alarm |
| (k) | LP | Light Pen Flip-Flops 1 and 2 |
| (l) | PETR | Photoelectric reader flip-flops 1, 2, 3 and 4 |
| (m) | Alarm Indicator | |

3. <u>Flip-Flop Registers</u>

   (a) MAR
   (b) PC
   (c) MBR
   (d) AC
   (e) LR

4. <u>Switches</u>

(a) Suppress Alarm
(b) Suppress Chime
(c) Automatic Restart
(d) Automatic Read-In
(e) Automatic Test
(f) Stop on Cycle Zero
(g) Stop on Cycle One
(h) Step
(i) Repeat
(j) Printer Input

5. <u>Toggle Switch Registers</u>

(a) TAC - Toggle switch accumulator
(b) TBR - Toggle switch buffer register
(c) TSS - Sixteen toggle switch storage registers

## APPENDIX B

### Operate Class Command Summary

| | | | |
|---|---|---|---|
| CLL | (0.8) | opr 100,000 | Clear left AC |
| CLR | (0.8) | opr 40,000 | Clear right AC |
| IOS | (0.8) | opr 20,000 | In-out stop |
| HLT | (1.8) | opr 30,000 | Halt |
| P7H | (0.8) | opr 7,000 | Punch 7 holes |
| P6H | (0.8) | opr 6,000 | Punch 6 holes |
| PNT | (0.8) | opr 4,000 | Print |
| R1C | (0.8) | opr 1,000 | Read 1 line |
| R3C | (0.8) | opr 3,000 | Read 3 lines |
| DIS | (0.8) | opr 2,000 | Display |
| SHR | (1.4) | opr 400 | Shift right |
| CYR | (1.4) | opr 600 | Cycle right |
| MLR | (1.3) | opr 200 | MBR $\longrightarrow$ LR |
| PEN | (1.1) | opr 100 | Read light pen |
| TAC | (1.1) | opr 4 | TAC ones $\rightarrow$ AC |
| COM | (1.2) | opr 40 | Complement AC |
| PAD | (1.4) | opr 20 | Partial ADD MBR and AC |
| CRY | (1.7) | opr 10 | Partial ADD carry digits and AC |
| AMB | (1.2) | opr 1 | AC $\rightarrow$ MBR |
| TBR | (1.2) | opr 3 | TBR $\rightarrow$ MBR |
| LMB | (1.3) | opr 2 | LR $\rightarrow$ MBR |

## APPENDIX C

### OPERATE CLASS COMMAND COMBINATION SUMMARY

| | | | | |
|---|---|---|---|---|
| opr | 140,000 | = | Clear AC Clear AC | (CLA) |
| opr | 31 | = | Cycle left | (CYL) |
| opr | 140,040 | = | Clear and complement AC | (CLC) |
| opr | 22,000 | = | Display | |
| opr | 160,000 | = | In out stop and AC cleared | |
| opr | 27,600 | = | Punch 7 holes, cycle AC right | |
| opr | 26,600 | = | Punch 6 holes, cycle AC right | |
| opr | 166,000 | = | Clear AC - Punch blank tape | |
| opr | 24,600 | = | Print and cycle AC right | |
| opr | 27,021 | = | Punch 7 holes and clear AC | |
| opr | 26,021 | = | Punch 6 holes and clear AC | |
| opr | 24,021 | = | Print and leave AC cleared | |
| opr | 141,000 | = | Clear AC and start PETR running | |
| opr | 1,031 | = | Start PETR running and cycle left | |
| opr | 1,600 | = | Start PETR running, cycle right | |
| opr | 163,000 | = | Clear AC and read 3 lines of tape | |
| opr | 161,000 | = | Clear AC and read 1 line of tape | |
| opr | 161,031 | = | Read 1 line of tape and cycle left | |
| opr | 161,600 | = | Read 1 line of tape and cycle right | |
| opr | 140,004 | = | TAC $\rightarrow$ AC | |
| opr | 30 | = | Full add MBR and AC | |
| opr | 140,022 | = | LR $\rightarrow$ AC | (LAC) |
| opr | 201 | = | AC $\rightarrow$ LR | (ALR) |
| opr | 22 | = | Partial add LR and AC | (LPD) |
| opr | 200 | = | Clear LR | (LRO) |
| opr | 32 | = | Full add LR and AC | (LAD) |
| opr | 140,023 | = | TBR $\rightarrow$ AC | (TBR) |

A-68266

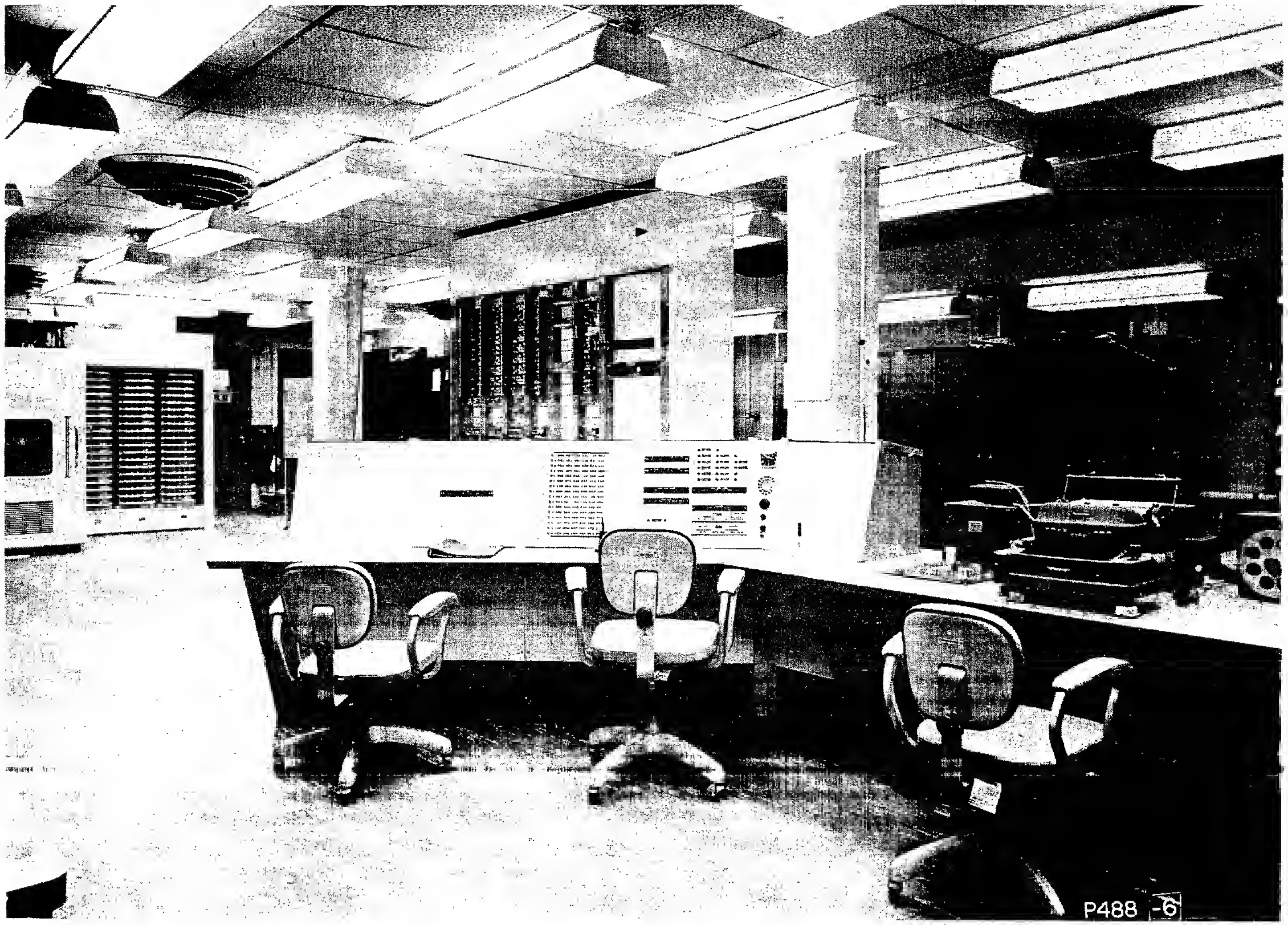


FIG. 1
TX-0 COMPUTER ROOM

A-68264

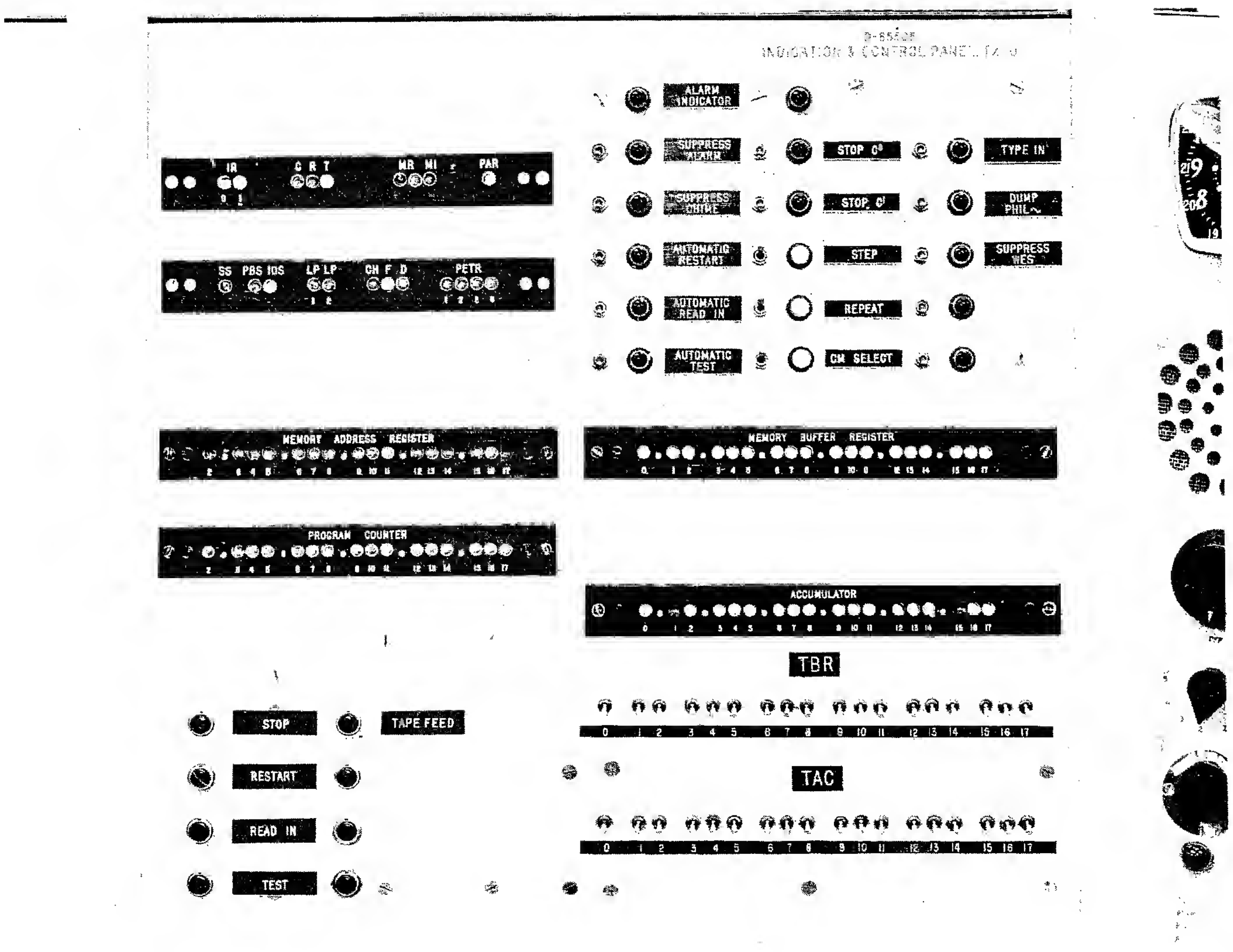


FIG. 2
TX-0 MAIN CONSOLE PANEL

A-68265

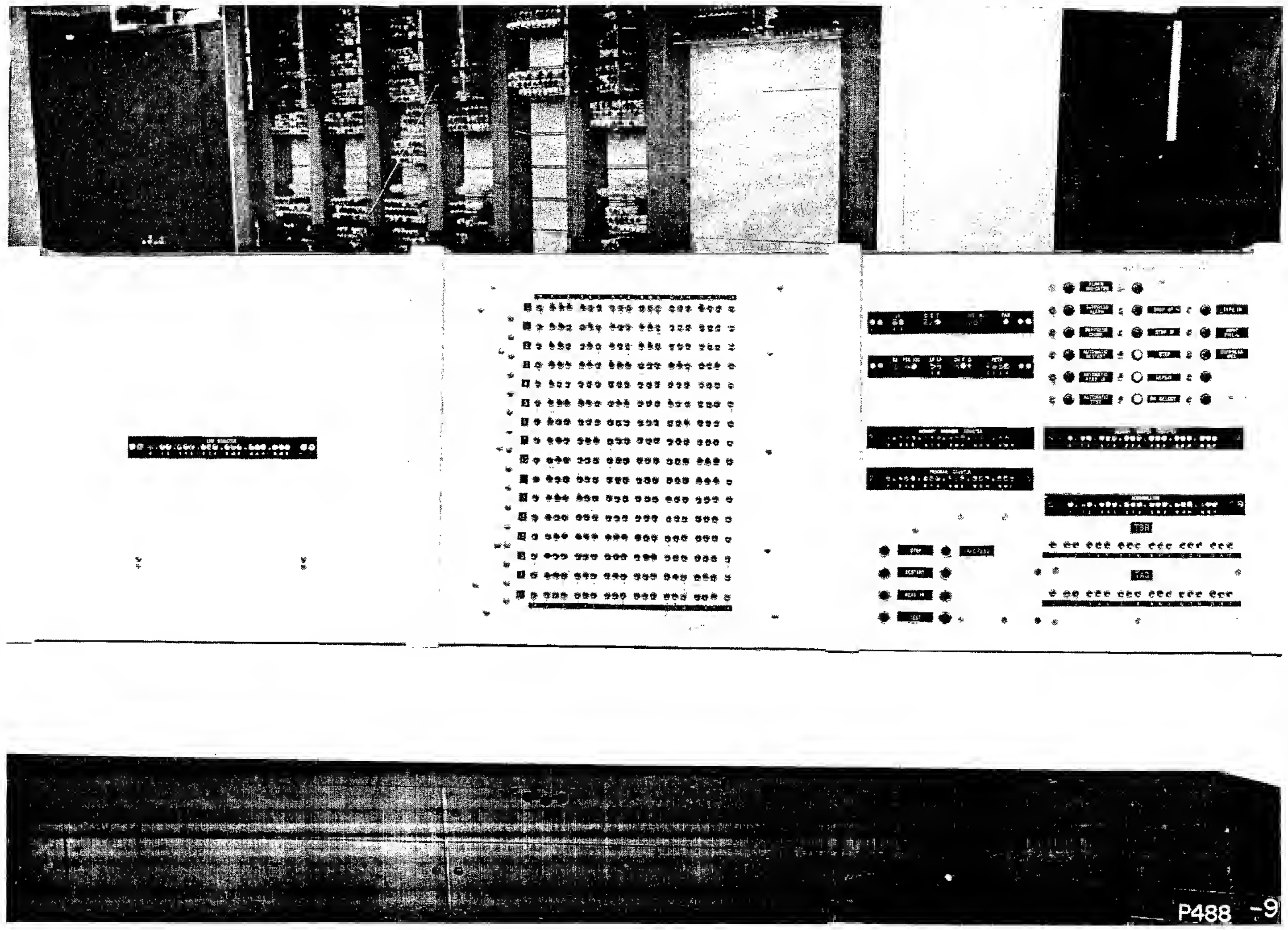


FIG. 3
TX-0 CONSOLE

A-68263

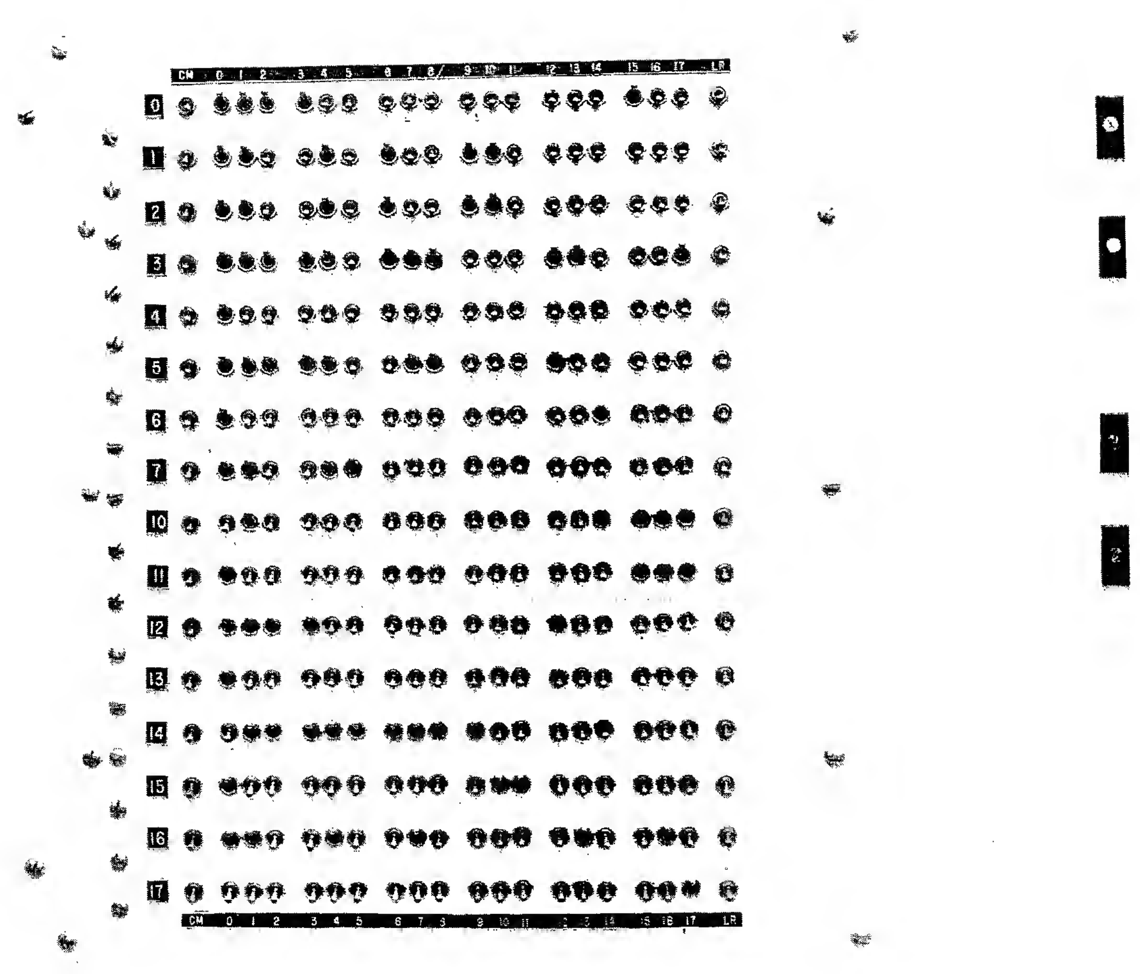
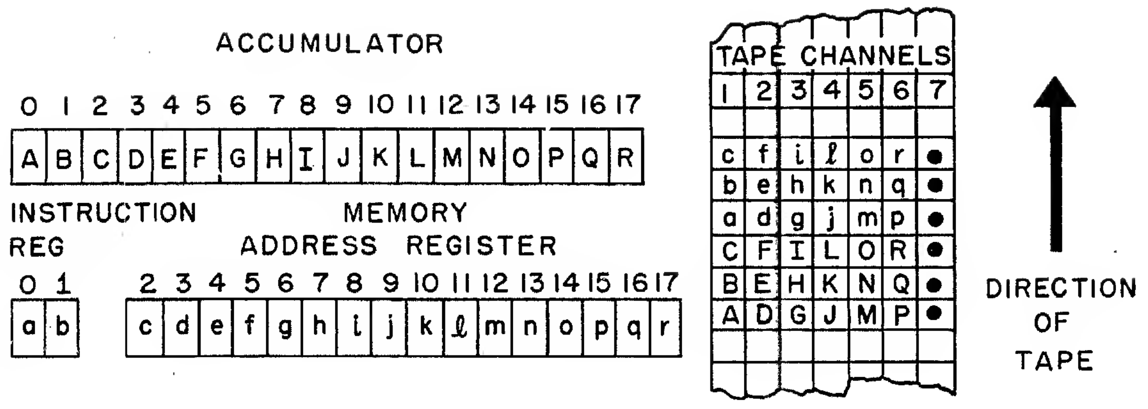
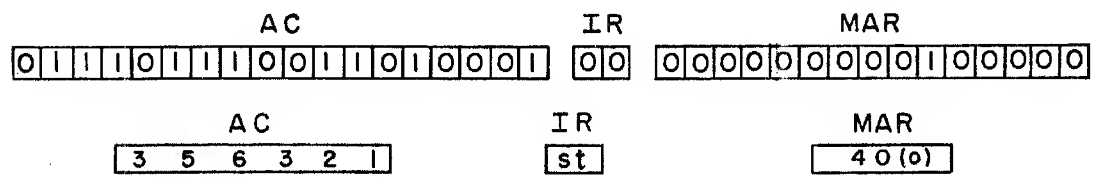
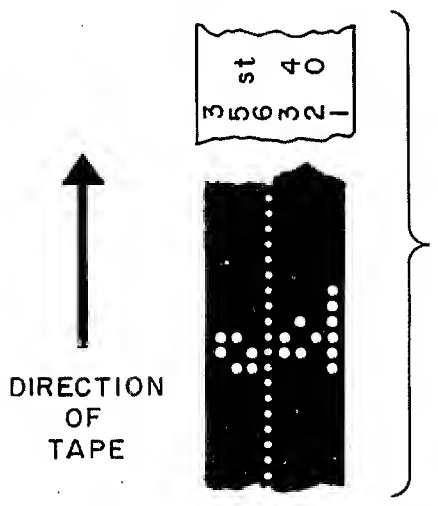


FIG. 4
TX-0 TOGGLE SWITCH STORAGE



EXAMPLE: STORE THE OCTAL WORD 356321 IN REGISTER 40 OCTAL



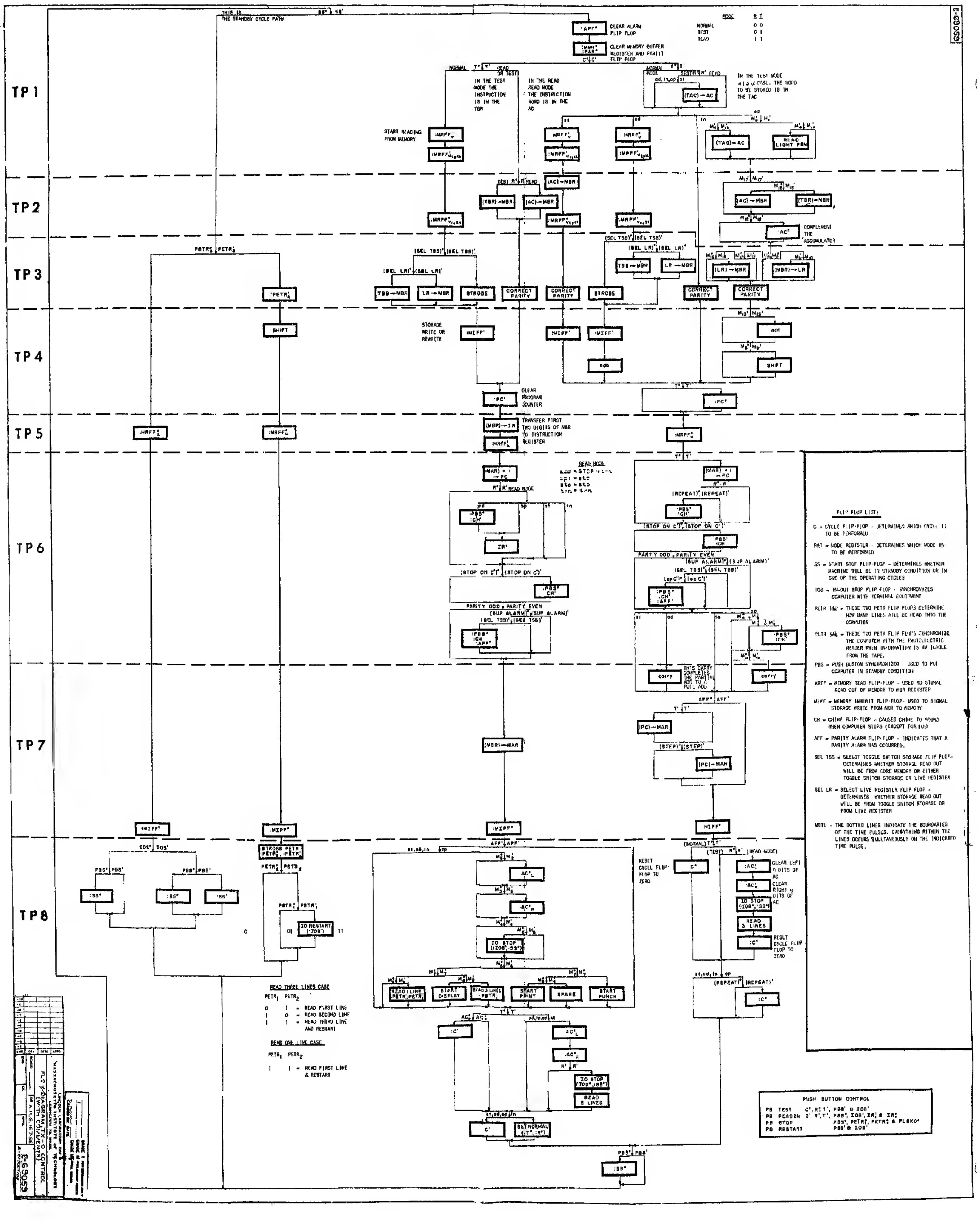


IF THE TAPE IS HELD SO THAT IT IS MOVING FROM LEFT TO RIGHT WITH THE SEVENTH HOLE NEXT TO THE BODY, THE DATA WORD AND STORE INSTRUCTION CAN BE READ OCTALLY IN HORZONTAL FASHION.

FIG. 5

TAPE LAYOUT FOR READ-IN MODE OF TX-0

A-68405



**READ THREE LINES CASE**

| PETR₁ | PETR₂ | | |
|---|---|---|---|
| 0 | 1 | = | READ FIRST LINE |
| 1 | 0 | = | READ SECOND LINE |
| 1 | 1 | = | READ THIRD LINE AND RESTART |

**READ ONE LINE CASE**

| PETR₁ | PETR₂ | | |
|---|---|---|---|
| 1 | 1 | = | READ FIRST LINE & RESTART |

**FLIP FLOP LIST:**

C = CYCLE FLIP-FLOP - DETERMINES WHICH CYCLE IS TO BE PERFORMED

R&T = MODE REGISTER - DETERMINES WHICH MODE IS TO BE PERFORMED

SS = START STOP FLIP-FLOP - DETERMINES WHETHER MACHINE WILL BE IN STANDBY CONDITION OR IN ONE OF THE OPERATING CYCLES

IOS = IN-OUT STOP FLIP-FLOP - SYNCHRONIZES COMPUTER WITH TERMINAL EQUIPMENT

PETR 1&2 = THESE TWO PETR FLIP FLOPS DETERMINE HOW MANY LINES WILL BE READ INTO THE COMPUTER

PETR 3&4 = THESE TWO PETR FLIP FLOPS SYNCHRONIZE THE COMPUTER WITH THE PHOTOELECTRIC READER WHEN INFORMATION IS AVAILABLE FROM THE TAPE.

PBS = PUSH BUTTON SYNCHRONIZER - USED TO PUT COMPUTER IN STANDBY CONDITION

MRFF = MEMORY READ FLIP-FLOP - USED TO SIGNAL READ OUT OF MEMORY TO MBR REGISTER

MIFF = MEMORY INHIBIT FLIP-FLOP - USED TO SIGNAL STORAGE WRITE FROM MBR TO MEMORY

CH = CHIME FLIP-FLOP - CAUSES CHIME TO SOUND WHEN COMPUTER STOPS (EXCEPT FOR IOS)

AFF = PARITY ALARM FLIP-FLOP - INDICATES THAT A PARITY ALARM HAS OCCURRED.

SEL TSS = SELECT TOGGLE SWITCH STORAGE FLIP FLOP - DETERMINES WHETHER STORAGE READ OUT WILL BE FROM CORE MEMORY OR EITHER TOGGLE SWITCH STORAGE OR LIVE REGISTER

SEL LR = SELECT LIVE REGISTER FLIP FLOP - DETERMINES WHETHER STORAGE READ OUT WILL BE FROM TOGGLE SWITCH STORAGE OR FROM LIVE REGISTER

NOTE - THE DOTTED LINES INDICATE THE BOUNDARIES OF THE TIME PULSES. EVERYTHING WITHIN THE LINES OCCURS SIMULTANEOUSLY ON THE INDICATED TIME PULSE.

**PUSH BUTTON CONTROL**

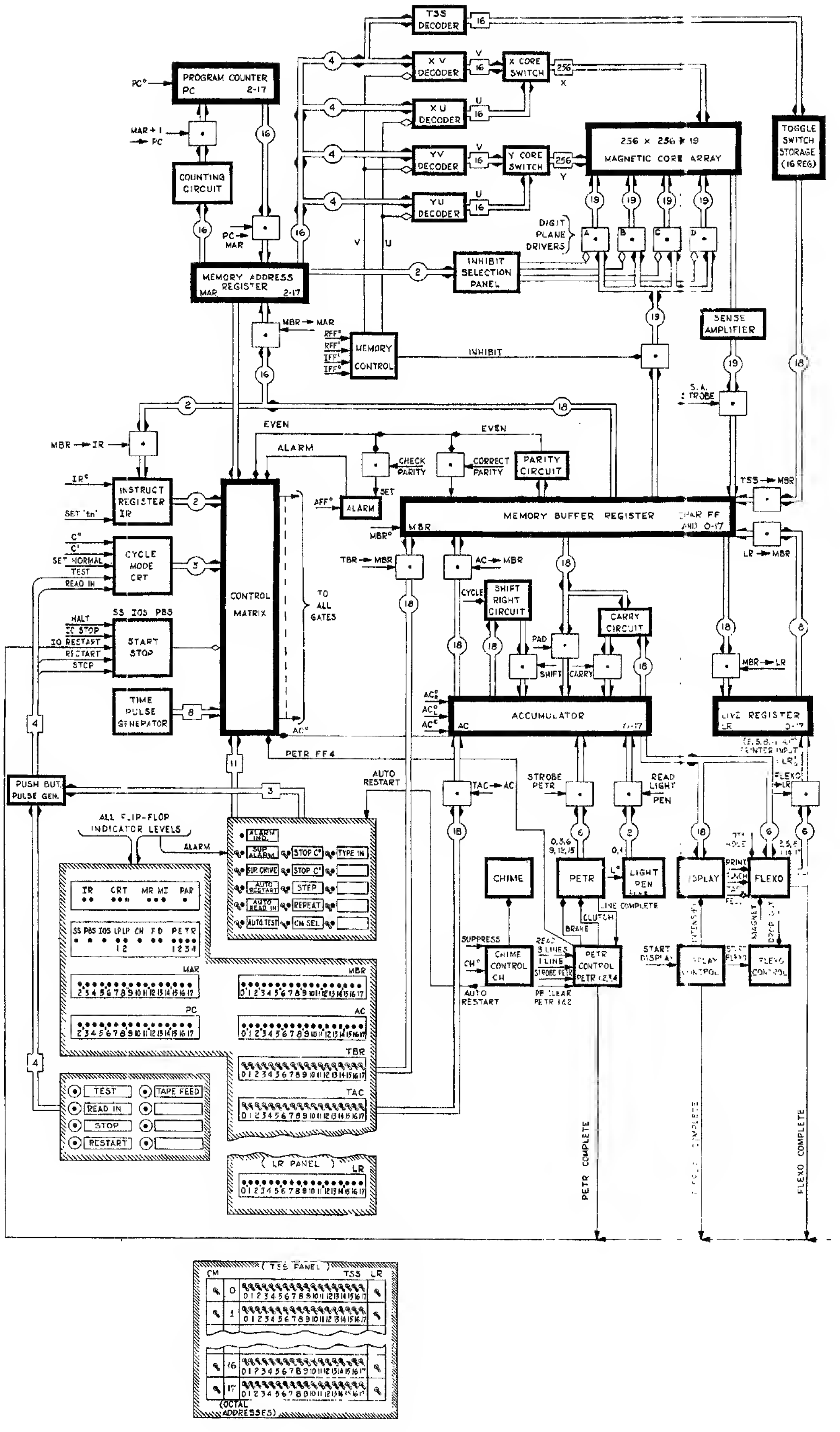
| | | |
|---|---|---|
| PB TEST | C°, R°, T', | PBS' & IOB' |
| PB READIN | C' R', T', | PBS°, IOB', IR₁ & IR₂ |
| PB STOP | | PBS°, PETR₁, PETR₂ & FLBKO° |
| PB RESTART | | PBS' & IOB' |

LINCOLN LABORATORY DIV. 6
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON 73, MASS.

FLOW DIAGRAM TX-0 CONTROL (WITH COMMENTS)

A.H.G. 12-7-56

E-69059



D-47243

FIG. 7

BLOCK DIAGRAM, TX-0